Assignment 1

a

Sorting the functions in ascending order:

$\log\log n < \log n < \sqrt{n} < n < n\log n < n^{\frac{3}{2}}$ -

$- < n^2 < n^2\log n < n^3 < 2^n < e^{n+1} < n!$

b

i $n^2 + 15n - 3 = \Theta(n^2)$

$f(n) = n^2 + 15n - 3$, $g(n) = n^2$

$\therefore f(n) \leq c\,g(n)$

$n^2 + 15n - 3 \leq c \cdot n^2$

$n^2 + 15n - 3 \leq cn^2 + 15n - 3$

$n^2 \leq cn^2$

This is True for $c = 1$ and $n = 1$

$\therefore n^2+15n-3 = O(n^2)$

again, for Big Omega,

$$f(n) \geq c \cdot g(n)$$

$$n^2+15-3 \geq c \cdot n^2$$

if we consider $c = 1$, so $n^2+15-3 \geq n^2$

its true.

$$\therefore n^2+15n-3 = \Omega(n^2$$

As n^2 is both upper and lower bound of

$n^2+15n-3$.

$$\therefore n^2+15n-3 = \Theta(n^2)$$

ii $4n^3 - 7n^2 + 15n - 3 = \Theta(n^3)$

first

$$4n^3 - 4n^2 + 15n - 3 \leq c \cdot n^3$$

$\Rightarrow 4n^3 - 4n^2 + 15n - 3 \leq c\,n^3 - 4n^2 + 15n - 3$

$\Rightarrow 4n^3 \leq c\,n^3$

for $c = 4$ the uppon function is true

$\therefore \quad 4n^3 - 7n^2 + 15n - 3 = O(n^3)$

again,

$$4n^3 - 7n^2 + 15n - 3 \geq cn^3$$

for $c = 1$

the upper function is true.

$\therefore \; 4n^3 - 7n^2 + 15n - 3 = \Omega(n^3)$

$\therefore$ n^3 is both upper and lower bound for that following function.

$\therefore \; 4n^3 - 7n^2 + 15n - 3 = \Theta(n^3)$

III $\quad T(n) = 4T(n/2) + n$

Here,

$a = 4$

$b = 2 \qquad$ so, $b^k = 2$

$c = 1 \qquad \therefore b^k < a$

$k = 1$

So, Applying the Master theorum the

Answe is $\theta\left(n^{\log_b a}\right)$ Here, $b = 2$, $a = 4$

$= \theta\left(n^{\log_2 2^4}\right) = \theta\left(n^{2\log_2 2^2}\right)$

$= \theta\left(n^2\right)$ (Ans)

(Proved)

iv) $T(n) = 2T(n/2) + n^3$

here, $a = 2$

$b = 2$ so, $b^k = 8$

$c = 1$ $\therefore b^k > a$

$k = 3$

$\therefore$ applying the Master theorum,

$\theta\left(n^k\right) = \theta\left(n^3\right)$ (Proved)

(v) $T(n) = T(n/4) + T(5n/8) + n$

first let us solve $\left(T(5n/8) + n\right)$ part here, $a = 1$

$b = \frac{8}{5}$ So, $b^k = \frac{8}{5}$

$c = 1$ $\therefore b^k > a$

$k = 1$

$\therefore$ Applying Master theorum:

$O(n^k) = \{O(n)$

now, $T(n) = T(n/4) + T(5n/8) + n$

$= T(n/4) + O(n)$

$= T(n/4) + n$

here,

$a = 1$

$b = 4$ so, $b^k = 4$

$c = 1$ $\therefore b^k > a$

$k = 1$

$\therefore$ Again Applying Mastere theorum:

$O(n^k) = (O(n))$ (proved)

vi $T(n) = T(n/3) + T(4n/9) + n$

~~fri~~ first lets solve $T(4n/9) + n$

$a = 1$

$b = \frac{9}{4}$ $\therefore$ $b^k = \frac{9}{4}$

$c = 1$ $\therefore b^k > 1$

$k = 1$

$\therefore$ Applying Master theorum: $O(n^k) = O(n)$

Again $T(n) = T(n/3) + O(n)$

$= T(n/3) = n$

$\therefore$

$a = 1$ so $b^k = 3$

$b = 3$

$c = 1$ $\therefore b^k > a$

$k = 1$

$\therefore$ Again Applying Master theorum: $O(n^k)$

$k = 1$ $\therefore$ $O(n)$ (proved)

c

1

for the 2nd loop,

for (j=1, j<=i, j++)

lets assume for the worst case the $i=n$, so the loop will run for n times

here time complexity is $\Theta(n)$

Now, for ($i=1$, $i<n$, $i*=2$)

here,

steps	i	
0	1	$=2^0$
1	2	$=2^1$
2	4	$=2^2$
3	8	$=2^3$
⋮		
for k steps k	2^k	

for worst case (i=n): $2^k = n$

$\Rightarrow \log_2 2^k = \log_2 n$

$\Rightarrow K = \log_2^n$

$\therefore$ for count++ the time complexity $= O(1)$

$\therefore$ Total time complexity $\left(O(n) * \log_2^n\right)$

$= O\left(n \log_2^n\right)$ (Ans)

2

$P = 3$

while $(P < n)$:

$P = P * P$

so,

steps	P	
0	3	$- 3^{2^0}$
1	9	$- 3^{2^1}$
2	81	$- 3^{2^2}$

⋮

[for ith steps] $i \rightarrow 3^{2^i}$

the worst will occur when $p = n$

$\therefore \quad 3^{2^i} = n$

$\Rightarrow \log_3 3^{2^i} = \log_3 n$

$\Rightarrow 2^i = \log_3 n$

$\Rightarrow \log_2 2^i = \log_2 \log_3 n$

$\Rightarrow i = \log_2 \log_3 n$

$\therefore$ Time Complexity $= O(\log_2 \log_3 n)$

(Ans)

d

In the following code,

first of all the initial array

size is lets consider : n

every time the array is divided into 3 parts

$\therefore$ sub array size $= \frac{n}{3}$

num of subarrays we are considering : 1

$\therefore$ the recursive relation:

$$T(n) = T(n/3) + O(1)$$

$$= T(n/3) + 1 \times n^0$$

here, $a = 1$, $b = 3$, $c = 1$, $k = 0$

$\therefore b^k = 1$, $b^k = a$

$\therefore$ Time complexity $= O(n^0 \log_3 n)$

$= O(\log_3 n)$

Another ~~App~~ approach

first,

Step	array size
0	$n = n/3^0$
1	$n/3 = n/3^1$
2	$n/9 = n/3^2$
$\vdots$	
K	$n/3^k$

So eventually at the worst case the array will become size of 1

$\therefore \quad \frac{n}{3^K} = 1$

$3^K = n$

$\log_3 3^K = \log_3 n$

$k = \log_3 n$

$\therefore$ Time complexity $= O(\log_3 n)$ (Ans)

2

a

```
def task4 (ar1 , ar2):
    for i in range (len (ar2)):
        Store = Search (ar1, ar2[i])
        print (store, end = ' ')

def Search (ar1, val):
    L = 0
    R = len (ar1) - 1
    M = 0
    while (L< =R):
        M = (L+R) // 2
        if (val = ~~A~~ ar1[M]):

            L+ = 1
            R+ = 1
            if M = len(ar1) - 1 :

                return R
            if val ! = ar1[M+1] :
                return M+1
```

```
        elif (val > ar1[M]):
            L = M+1

        else :
            R = M - 1

    if L == (len (ar1)):
        return L
    elif R == -1:
        return 0
    else :
        return L


# Tester class
ar1 = [1,1,2,2,5]
ar2 = [3,1,4,1,2]
task4 (ar1, ar2)
```

b

For the "Search" function,

~~I~~ am using Binary ~~Bina~~ Search method to searching elements,

~~I~~ In binary search,

The number of subarrays $= 2$

working with subarray $= 1$

So the time complexity for binary search:

step	array size
1	$\frac{n}{2} = \frac{n}{2^1}$
2	$\frac{n}{4} = \frac{n}{2^2}$
3	$\frac{n}{8} = \frac{n}{2^3}$
⋮	
i	$\frac{n}{2^i}$

~~So after~~ n: $\frac{n}{2^i} = 1 \Rightarrow n = 2^i \Rightarrow \log_2 n = \log_2 2^i$

$\Rightarrow i = \log_2 n$

$\therefore$ time complexity $= O(\log_2 n)$

again for "task 4" function

the for loop is running for n times

so here the time ~~coplex~~ complexity

$= O(n)$

$\therefore$ Total time complexity $= n \times \log_2 n$

$= O(n \log_2 n)$ (Ans)